The plotStuff Graphics Post Processor for Overture
User Guide, Version 1.00

Bill Henshaw

Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551
henshaw@llnl.gov
http://www.llnl.gov/casc/people/henshaw
http://www.llnl.gov/casc/Overture

May 14, 2004

UCRL-MA-138730

Abstract: This document explains how to use `plotStuff`, a program for plotting results from Overture programs. `plotStuff` reads data-base files created by Overture that contain information about the grid and solutions, a "show file". `plotStuff` has features for plotting contours, cutting planes, streamlines, iso-surfaces and grids. It can also be used to plot various derived quantities such as derivatives of the solution components. This document describes the class `DerivedFunctions` which is used to compute these derived quanties.

Contents

1 Introduction

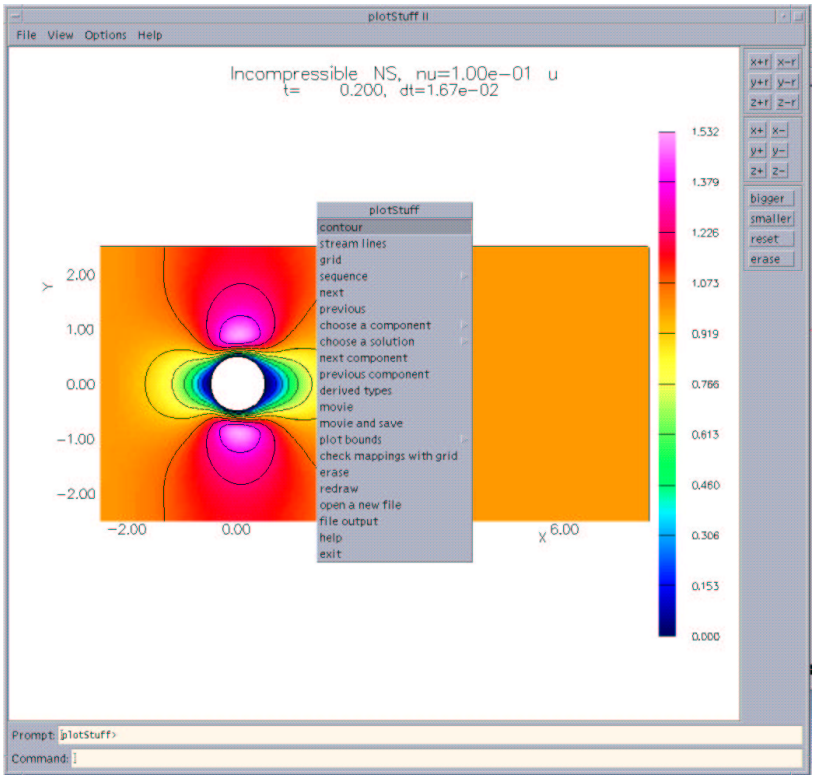

Figure 1: plotStuff is used to display results from an Overture data-base file (‘show file’) such as those created by the **OverBlown** flow solver

The `plotStuff` program, usually found in `Overture/bin/plotStuff` can be used to plot results found in “show files” or “grid files” created by Overture programs.

A “show file” contains a sequence of **frames**, each frame will normally contain a “solution” (i.e. a grid function defined on a grid). In the moving grid case each frame may also contain a new grid. Each solution will consist of a number of `components`, such as the velocity and pressure. A show file may also contain one or more **sequence**, which is an array of values such as the maximum residual over time.

Some of the features of `plotStuff` include

- plot contours of various flavours, plot streamlines, plot grids.
- show a movie of a time sequence of frames; save hard-copies for making mpeg movies.
- plot “derived types” from variables in the show file, such as plotting the vorticity given the velocity or plotting derivatives of the variables.

The `plotStuff` program is basically a front end to the features available in other classes such as `GenericGraphicsInterface`[2], Ogshow[1], `ShowFileReader`[1] and the `DerivedFunctions` class.

2 Typical usage

The typical usage of `plotStuff` is to start the program up by typing “`plotStuff fileName.show`” or just “`plotStuff fileName`” since the “.show” is optional. Here `fileName.show` is the name of a show file such as one created by **OverBlown**[3]. Now one can choose `contour` to plot contours. At this point a new menu appears, this is the contour plotter menu. By choosing `exit` one returns to the main `plotStuff` menu. Choosing a new solution, or typing `next` or `previous` will plot contours of a different solution. One can also plot a different component. At this point one could choose `grid` or ‘`stream lines`’ to plot the grid or plot streamlines. Note that the contour plot will not be erased by default. This

allows multiple things to be plotted at once. One can choose `erase` to remove the contour plot before plotting other things. By choosing the '`derived types`' menu item on can build new components from existing ones. For example the vorticity or divergence can be created from a velocity vector. After these new derived types have been created they will appear in the `plotStuff` component menus.

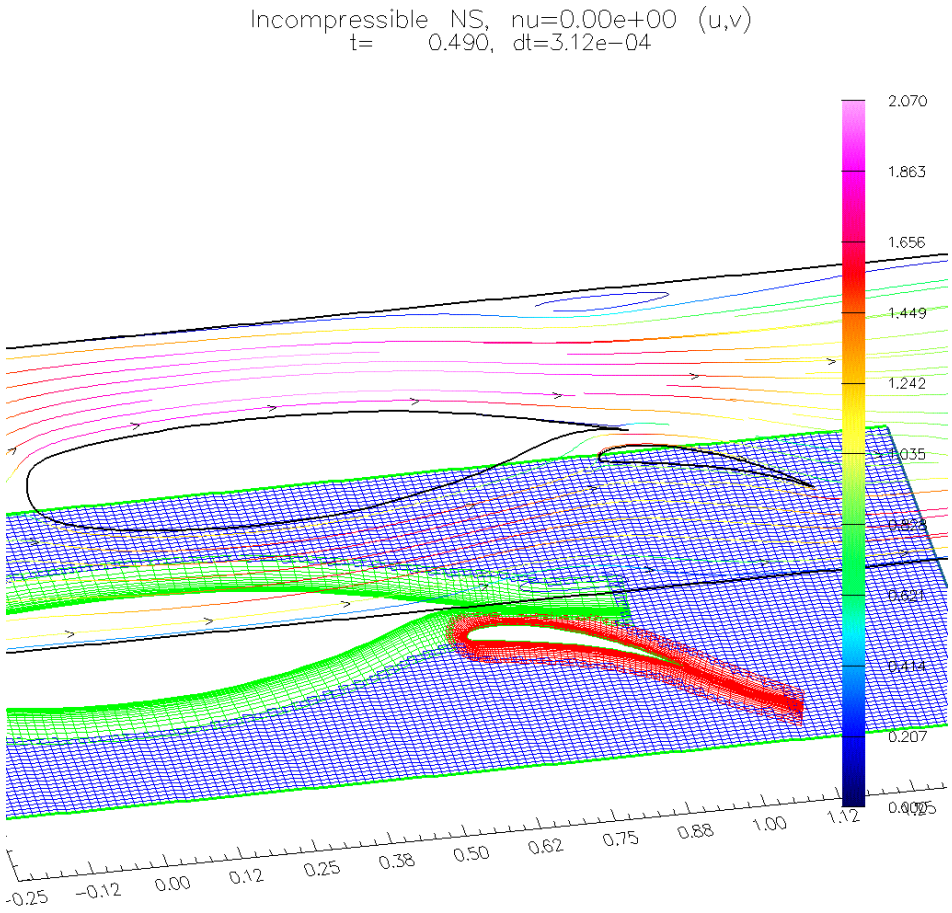

Figure 2: plotStuff can be used to plot more than one quantity on the same figure; here streamlines and a grid are plotted. This plot used an option in the grid-plotter to change the height of the grid. This computation was performed by the **OverBlown** solver.

3 The plotStuff menu items

Here is a desciption of the menu options available for plotStuff.

contour : plot contours.

stream lines : plot streamlines.

grid : plot the grid.

sequence : plot a sequence.

next : plot solutions from the next frame.

previous : plot solutions from the previous frame.

choose a component : plot a different component.

choose a solution : choose a different frame.

next component : plot the next component.

previous component : plot the previous component.

derived types : build new components as functions of the old ones, such as the vorticity from the velocity. Once derived types have been created they will appear in the plotStuff component menus.

movie : plot a number of frames in a row.

movie and save : plot frames and save each one as a hard copy.

plot bounds : change the manner in which the plot bounds are determined.

set plot bounds : specify bounds for plotting.

use default plot bounds : use default plot bounds.

check mappings with grid : a debugging option, check validity of mappings after they have been read in from a data-base file.

erase : erase anything in the window.

redraw : redraw the screen.

open a new file : open a new show file for reading.

file output : output results to a file.

help : print a short help list.

exit : exit this menu and continue on (same as 'continue').

4 Derived quantities

Various derived quantities can be plotted with `plotStuff`. The class `DerivedFunctions`, described in more detail in the next section, is used to define derived functions for plotting with `plotStuff`. For example one can compute the vorticity given some velocity components. One can also differentiate components. There is also a way for users to define new derived quantities by editing the functions in the `userDefinedDerivedFunction.C` file and re-linking this with `plotStuff`.

Here are some of the derived functions we can create

x,y,z,xx,yy,zz,laplacian : compute first and second derivatives of any components,

$$\begin{aligned}
\text{x} &= \partial_x u \\
\text{y} &= \partial_y u \\
\text{z} &= \partial_z u \\
\text{xx} &= \partial_{xx} u \\
\text{yy} &= \partial_{yy} u \\
\text{zz} &= \partial_{zz} u \\
\text{laplacian} &= \Delta u = \nabla \cdot \nabla u = \partial_x^2 u + \partial_y^2 u + \partial_z^2 u
\end{aligned}$$

vorticity The same as zVorticity, see below. In 2D this is the only non-trivial component of the vorticity.

xVorticity, yVorticity, zVorticity Components of the curl of the velocity, $\nabla \times \mathbf{u}$,

$$\begin{aligned}
\text{xVorticity} &= \partial_z v - \partial_y w \\
\text{yVorticity} &= \partial_x w - \partial_z u \\
\text{zVorticity} &= \partial_y u - \partial_x v
\end{aligned}$$

divergence The dilatation or divergence of the velocity, $\nabla \cdot \mathbf{u}$,

$$\text{divergence} = \partial_x u + \partial_y v + \partial_z w$$

mach Number The ratio of the local magnitude of velocity to the local speed of sound:

$$\begin{aligned}
\text{mach Number} &= \frac{\sqrt{u^2 + v^2 + w^2}}{a} \\
a &= \sqrt{\gamma R_g T} \qquad \text{if T is defined} \\
a &= \sqrt{\gamma p/\rho} \qquad \text{if p and } \rho \text{ are defined}
\end{aligned}$$

temperature The temperature as a function of the pressure, density and γ,

$$\text{temperature} = \frac{\gamma p}{\rho}$$

speed The magnitude of the velocity, $\sqrt{\mathbf{u} \cdot \mathbf{u}}$,

$$\text{speed} = \sqrt{u^2 + v^2 + w^2}$$

schlieren Use this quantity with the "gray" colour table to generate a *Schlieren* image. For hardcopy, save the image as "gray scale" (from "output format menu" in the "file" pull-down menu). The exponential in the formula below acts to enhance fine scale structure.

$$\begin{aligned}
\text{schlieren} &= \alpha \exp(-\beta s(\mathbf{x})) \qquad (\alpha = 1,\ \beta = 15) \\
s(\mathbf{x}) &= (\tilde{s} - \min \tilde{s})/(\max \tilde{s} - \min \tilde{s}) \qquad (s \in [0,1]) \\
\tilde{s}(\mathbf{x}) &= \sqrt{\rho_x^2 + \rho_y^2 + \rho_z^2}
\end{aligned}$$

The quantities α and β can be changed using the `schlieren parameters` menu.

minimumScale, r1MinimumScale, r2MinimumScale, r3MinimumScale Local measures of the minimum scale. Where the minimun scale is large, the flow is less well resolved. The quantities r1MinimumScale, r2MinimumScale, and r3MinimumScale are meant to measure how well the flow is resolved along each coordinate direction. See the discussion in the OverBlown reference guide. I think the definition of r1MinimumScale etc. could be simplified to the second expression below (Anders?)

$$\text{minimumScale} = \|\Delta \mathbf{x}\| \left[\sum_{m,n} \frac{\partial u_m}{\partial x_n} \frac{1}{\nu} \right]^{1/2}$$

$$\begin{aligned} \text{r[i]MinimumScale} &= \Delta r_i \left[\sum_{n} \frac{\partial x_n}{\partial r_i} \right]^{1/4} \left[\sum_{m,n} \frac{\partial u_m}{\partial x_n} \frac{\partial x_n}{\partial r_i} \frac{1}{\nu} \right]^{1/2} \\ &= \Delta r_i \left[\sum_{m} \frac{\partial u_m}{\partial r_i} \frac{1}{\nu} \right]^{1/2} \end{aligned}$$

user defined This option can be used to access any user defined derived functions; these are defined in the file `userDefinedDerivedFunction.C`.

The components u,v,w etc. are determined by looking for the following variables in the show file,

uComponent : the component number of x-component of the velocity u.

vComponent : the component number of y-component of the velocity v.

wComponent : the component number of z-component of the velocity w.

pressureComponent : the component number of the pressure p.

temperatureComponent : the component number of the temperature T.

densityComponent : the component number of the density ρ.

For example, **OverBlown** saves the incompressible flow components with `pressureComponent=0`, `uComponent=1`, `vComponent=2`, and `wComponent=3`. See the **OverBlown** file `saveShow.C` for an example of saving this extra info in the show file.

Other variables that may be required in the show file are

gamma : ration of specific heats (if constant).

Rg : the gas constant.

nu : kinematic viscosity.

5 The DerivedFunctions class

Here is a description of the `DerivedFunctions` class.

5.1 constructors

DerivedFunctions(ShowFileReader & showFileReader_)

Description: Derive new functions from old ones. This class is used by plotStuff to create derived quantities such as vorticity, mach-number, derivatives etc.

5.2 getASolution

int
getASolution(int & solutionNumber,
MappedGrid & cg,
realMappedGridFunction & u)

5.3 getASolution

int
getASolution(int & solutionNumber,
CompositeGrid & cg,
realCompositeGridFunction & u)

5.4 getComponent

int
getComponent(int & c, const aString & cName)

Access: protected.

Description: Look for a component number in the data base.

c (input): c==-2 first time through. If not found c==-1 on output.

Return value: 0=success, 1=not found.

5.5 computeDerivedFunctions

int
computeDerivedFunctions(realCompositeGridFunction & u)

Access: protected.

Description: This function will compute the derived types and add these as new components to the end of the grid function u. This function is normally called after reading a new solution from the show file. It used the showFileReader pointer to get info about the current frame.

u (input/output): On input u should contain valid values for the default components. On output u will be redimensioned to hold th extra derived components.

5.6 add

int
add(int derivative, const aString & name_,
int n1 =0,
int n2 =0)

Description: Add a new entry to the list of derived functions.

derivative (input) : derived type code.

name_ (input) : component name for this entry.

n1,n2 (input) : extra info to save in the derived array.

5.7 remove

int
remove(int i)

Description: remove an entry from the list of derived types.

5.8 update

int
update(GenericGraphicsInterface & gi,
int numberOfComponents,
aString *componentNames)

Description: update current list of derived grid functions

5.9 setupUserDefinedDerivedFunction

int
setupUserDefinedDerivedFunction(GenericGraphicsInterface & gi,
int numberOfComponents,
aString *componentNames)

Description: User defined derived functions. This function is used to setup and define a derived function. The function getUserDefinedDerivedFunction is called to actually evaluate the derived function. Choose the "user defined" option from the DerivedFunctions::update options to have this routine called.

Notes:
- To to add a new user defined derived-function edit this file (Ogshow/userDefinedDerivedFunction.C) and rebuild the plotStuff executable using the new version. This can be done by rebuilding the Overture library in place, or by copying the files plotStuffDriver.C, plotStuff.C and userDefinedDerivedFunction.C to a new location and building a separate version of plotStuff.

Return values: 0=success, non-zero=failure.

5.10 getUserDefinedDerivedFunction

int
getUserDefinedDerivedFunction(int index,
int indexOut,
const aString & name,
const int numberOfComponents,
realCompositeGridFunction & uIn,
realCompositeGridFunction & uOut,
bool & interpolationRequired)

Description: Assign a user defined derived function.

index (input): the index of this derived function (derived functions have index=0,1,2,...)

indexOut (input): fill in this component of uOut (indexOut=index+numberOfComponents)

name (input): the name associated with the derived function

uIn : the solution read from the show file

uOut : fill the derived function into this composite grid function

Return values: 0=success, non-zero=failure.

References

[1] W. HENSHAW, *Ogshow: Overlapping grid show file class, saving solutions to be displayed with plotStuff, user guide*, Research Report UCRL-MA-132235, Lawrence Livermore National Laboratory, 1998.

[2] ——, *Plotstuff: A class for plotting stuff from Overture*, Research Report UCRL-MA-132238, Lawrence Livermore National Laboratory, 1998.

[3] ——, *OverBlown: A fluid flow solver for overlapping grids, user guide*, Research Report UCRL-MA-134288, Lawrence Livermore National Laboratory, 1999.

Index